Fingerprint Optimization Manual

Spencer Evans-Cole

July 30, 2024

Contents

1 Introduction

1.1 Problem Statement

The development of interatomic potentials via machine learning on first principles results is driven by the need to accurately and quickly analyze atomic environments. When encoding atomic environments for neural networks, we use the following two body or radial fingerprint

$$F_n = \sum_{j \neq i} \left(\frac{r_{ij}}{r_e} \right)^n e^{-\alpha_n \frac{r_{ij}}{r_e}} f_c \left(\frac{r_c - r_{ij}}{\Delta r} \right) S_{ij},$$

and three body or bond fingerprint

$$G_{m,k} = \sum_{\alpha_1 \geq \alpha_2 \geq \cdots \geq \alpha_m} \left(\frac{n!}{n_x! n_y! n_z!} \sum_j \frac{r_{ij}^{\alpha_1} r_{ij}^{\alpha} 2 \cdots r_{ij}^{\alpha} m}{r_{ij}^m} e^{-\beta_k \frac{r_{ij}}{r_e}} f_c \left(\frac{r_c - r_{ij}}{\Delta r} \right) \right)^2,$$

as detailed in [2]. Given a set of atomic environments, this software generates a series of different fingerprints and by analysis of this data, it selects the most optimal ones which demonstrate a reduced validation error and computation time in our model.

1.2 About this Document

This document is meant to explain how to use the fingerprint optimization software, and how it works. Section 2 discusses installation, the basic usage of the software, a general overview of how the code works, and possible issues one may face when running the software. The remaining sections of the document discuss the mathematical and practical implementations of the software.

To understand how to use this software one must know how to use a Unix operating system and basic usage of make for C++ compilation. To understand the mathematics of the program one must have an understanding of basic algebra, and fundamentals of linear algebra would be preferred but not required.

We also note that in the software and this document α_k and β_k are used interchangeably as there is a discrepancy in notation between the equations and input file to the calibration software.

1.3 Acknowledgements

This material is based upon work supported by the National Science Foundation under Grant No. DMR-2348712 . Any findings, opinions, conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect views of the National Science Foundation.

2 Installation and Usage

2.1 Installation

This software requires Armadillo [3], Open BLAS, Intel MKL, and Open MP. All of these packages should be installed via the makefile. To begin the installation process clone the GitHub repo by running the command

```
git clone https://github.com/sec05/Fingerprint-Optimization.git
```

Alternatively, one could clone via ssh or download a .zip of the software from GitHub.

Once downloaded, access the directory via `cd Fingerprint-Optimization`. From here we must install three modules via:

```
module load intel-oneapi-mkl
module load intel-oneapi-mpi
module load openblas
```

We now must install Armadillo and build the program. Decide on the director you wish to install Armadillo to and run the command

```
make -j 40 -f Makefile.hpc -e ARMA_DIR=/path/to/install
```

Assuming there are no errors you should see a `fingerprint-optimizer` file in your directory.

2.2 Input File

To use the software one must create an input file of the form

```
# Any lines with "#" will be ignored!
Input File:
Bi.nn
Output File:
# comment here
Bi.DCUR.out
Verbose:
1
Output Selections:
1
Radial Blocks:
100
Alpha Upper Bound:
10
Alpha Lower Bound:
0
ks:
```

```
125
Alpha_k Upper Bound:
10
Alpha_k Lower Bound:
0
Selections:
200
Selection Method:
2
Output Radial Blocks:
1
Output Bond Size:
10
```

Note that lines that start with "`#`" will be ignored and that the order of the properties can be in any order. Additionally, leading and trailing white space on each line will be ignored.

2.3 Usage

Assuming the software has been installed and has an input file called `opt.in`, we can invoke the software by running

```
./fingerprint_optimizer -in optimizer.in
```

After running the program, you will get an optimized output file for the calibration software in the location you specified in the input file. Additionally the files, `Optimizer Output/optimizer.in.opt`, and
`Optimizer Output/optimizer.in.optv` will be created, but they can be deleted after the program runs. The file `Optimizer Output/optimizer.in.opt` is the input file fed into the calibration software to generate all of the different fingerprints, and `Optimizer Output/optimizer.in.optv` contains a list of every generated fingerprint and their parameters.

If selections is set to 1, the file `optimizer.out.sel` will be created which contains all of the fingerprint parameters the program selected ranked by importance, for radial fingerprints this is a value of n and α_n for bond fingerprints this is a value of k, m, and β_k.

We generate values radial fingerprints by blocks and bond fingerprints by values. One block of radial fingerprints is one set of α_n. For example, if $o = -1$ and $n = 3$ then one block of radial fingerprints is 5 values of α_n. To change the values of o and n one must change the initial input file.

For both radial and bond values of α_n and β_k respectively, we generate a step size such that the lower bound plus the step size times the number of α_n or β_k respectively equal the input values. Verbose mode will print out the specific step size and number of values generated.

The amount of selections is the amount of columns we wish to sample from our matrix. We recommend having selections much larger than the desired

number of fingerprints, and for the selection method to be set to 2 which is the method detailed in 4.5. If the selection method is set to anything else the first k bond fingerprints will be selected.

3 Linear Algebra Prerequisites

3.1 Matrices and Vectors

Definition 3.1.1 (Matrix). A *matrix* is a grid of m rows and n columns of real numbers. More precisely, let $m, n \in \mathbb{Z}^+$, and for $1 \leq i \leq m$ and $1 \leq j \leq n$ define $a_{ij} \in \mathbb{R}$ as an element of a $m \times n$ matrix A denoted as

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}$$

since the members of A are real numbers we say $A \in \mathbb{R}^{m \times n}$.

Definition 3.1.2 (Matrix Transpose). Let $A \in \mathbb{R}^{m \times n}$, then the *matrix transpose* of A is a matrix $B \in \mathbb{R}^{n \times m}$ such that for $1 \leq i \leq n$ and $1 \leq j \leq m, b_{ij} = a_{ji}$. Commonly B is denoted A^T.

Definition 3.1.3 (Vector). A *vector* is a matrix of m rows and one column denoted as

$$v = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_m \end{bmatrix}$$

since the members of v are real numbers we say $v \in \mathbb{R}^m$.

Remark 3.1.4. Commonly, we think of a matrix as a list of vectors. Let $a_1, \ldots, a_n \in \mathbb{R}^m$ then we write a matrix $A \in \mathbb{R}^{m \times n}$ as

$$\begin{bmatrix} a_1 & \cdots & a_n \end{bmatrix}$$

Definition 3.1.5 (Square Matrix). A matrix $A \in \mathbb{R}^{m \times n}$ is a *square matrix* if and only if $m = n$.

Definition 3.1.6 (Diagonal Matrix). A square matrix $A \in \mathbb{R}^{n \times n}$ is a *diagonal matrix* if and only if for $1 \leq i, j \leq n$ and $i \neq j, a_{ii} \in \mathbb{R}$ and $a_{ij} = 0$.

Definition 3.1.7 (Identity Matrix). A diagonal matrix $I \in \mathbb{R}^{n \times n}$, is an *identity matrix* if and only if it has only ones on the diagonal.

Definition 3.1.8 (Trace). We define the trace of a matrix $A \in \mathbb{R}^{m \times n}$ as

$$\mathrm{Tr}(A) = \sum_{i=1}^{n} a_{ii}.$$

Definition 3.1.9 (Outer Product). Let $u \in R^m$ and $v \in \mathbb{R}^n$ then the *outer product* of u and v is defined as

$$uv^T = \begin{bmatrix} u_1v_1 & u_1v_2 & \cdots & u_1v_n \\ u_2v_1 & u_2v_2 & \cdots & u_2v_n \\ \vdots & \vdots & \vdots & \vdots \\ u_mv_1 & u_mv_2 & \cdots & u_mv_n \end{bmatrix}$$

Definition 3.1.10 (Matrix-Vector Product). Let $A \in \mathbb{R}^{m\times n}$ and $x \in R^n$ then the *matrix vector product*, Ax is a vector $b \in R^m$ where for $1 \leq i \leq m$ has entries b_i expressed as

$$b_i = \sum_{j=1}^{n} a_{ij}x_j$$

Definition 3.1.11 (Matrix-Matrix Product). Let $A \in \mathbb{R}^{m\times n}$ and $B \in \mathbb{R}^{n\times p}$, then the *matrix-matrix product*, AB is a matrix $C \in \mathbb{R}^{m\times p}$ where for $1 \leq i \leq m$ and $1 \leq j \leq p$ the entries of C are defined as

$$c_{ij} = \sum_{k=1}^{n} a_{ik}b_{kj}$$

Remark 3.1.12 (Matrix-Matrix Product via Outer Products). Let $A \in \mathbb{R}^{m\times n}$ and $B \in \mathbb{R}^{n\times p}$, then the product AB can be expressed as

$$AB = \sum_{i=1}^{n} a_i b_i^T$$

where a_i is the ith column of A and b_i^T is the ith row of B.

Definition 3.1.13 (Matrix Inverse). Let $A \in \mathbb{R}^{m\times n}$, then the *matrix inverse* of A if a matrix $A^{-1} \in \mathbb{R}^{n\times m}$ such that $AA^{-1} = I$ and $A^{-1}A = I$. Note that it is not guaranteed for A^{-1} to exist.

3.2 Inner Products and Norms

Definition 3.2.1 (Matrix Norm). Let $A \in \mathbb{R}^{m\times n}$ and $p \in \mathbb{Z}^+$ then we define a *matrix norm* as

$$\|A\|_p = \sup_{x\neq 0} \frac{\|Ax\|_p}{\|x\|_p}.$$

Definition 3.2.2 (Frobenius Norm). Let $A \in \mathbb{R}^{m\times n}$ then we define the *Frobenius norm* of A to be

$$\|A\|_F = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n} |a_{ij}|^2}.$$

Definition 3.2.3 (Vector Norm). Let $u \in \mathbb{R}^n$ and $p \in \mathbb{Z}^+$ then we define a *vector norm* as

$$\|u\|_p = \left(\sum_{i=1}^{n} |u_i| \right)^{1/p}.$$

Definition 3.2.4 (Inner Product). Given two vectors u and v in $\mathbb{R}^n$ we define the *inner product* between u and v as

$$u^T v = \begin{bmatrix} u_1 & \cdots & u_n \end{bmatrix} \begin{bmatrix} v_1 \\ \vdots \\ v_n \end{bmatrix} = u_1 v_1 + \cdots + u_n v_n = \sum_{i=1}^{n} u_i v_i$$

Definition 3.2.5 (Orthogonal Vector). Vectors u and v in $\mathbb{R}^n$ are *orthogonal* if and only if $u^T v = 0$. If $\|u\|_2 = 1$ and $\|v\|_2 = 1$ then u and v are *orthonormal*.

Theorem 3.2.6. A matrix $U \in \mathbb{R}^{m \times n}$ is orthonormal if and only if $UU^T = I$ and $U^T U = I$.

3.3 Subspaces

Definition 3.3.1 (Subspace). A *subspace* of $\mathbb{R}^n$ is a set of vectors W such that:

1. $0 \in W$.
2. If $u \in W$ and $v \in W$ then $u + v \in W$ for all $u \in W$ and all $v \in W$.
3. If $u \in W$ and $c \in \mathbb{R}$ then $cu \in W$ for all $u \in W$.

Definition 3.3.2 (Column Space). Let $A \in \mathbb{R}^{m \times n}$, the *column space* of A denoted $\text{col}(A)$ is a subspace defined by the span of the columns of A. Additionally, we define the row space of A to be $\text{col}(A^T)$ denoted $\text{row}(A)$.

Definition 3.3.3 (Rank). Let $A \in \mathbb{R}^{m \times n}$, then the rank of A denoted $\text{rank}(A)$ is defined to be the dimension of $\text{col}(A)$.

3.4 Projections

Definition 3.4.1 (Projection Matrix). A square matrix $P \in \mathbb{R}^{n \times n}$ is a *projection matrix* if and only if $P^2 = P$.

Definition 3.4.2 (Orthogonal Projector). A projection matrix $P \in \mathbb{R}^{n \times n}$ is an orthogonal projector if and only if $P = P^T$.

Remark 3.4.3. Let $U \in \mathbb{R}^{m \times n}$ be an orthonormal matrix, then the orthogonal projector onto $\text{col}(U)$ is given by $P = UU^T$.

Theorem 3.4.4 (Arbitrary Orthogonal Projector). Let $A \in \mathbb{R}^{m \times n}$ such that A^{-1} exists, then the orthogonal projector onto $\text{col}(A)$ is $P = A(A^T A)^{-1} A^T$.

Remark 3.4.5 (Moore-Penrose Inverse). Often times $(A^T A)^{-1} A^T$ is denoted as A^+, the *Moore Penrose inverse of A*. Then the orthogonal projector onto $\text{col}(A)$ can be written as $P = AA^+$.

3.5 Eigenvalue Problems

Definition 3.5.1 (Eigenvalues and Eigenvectors). Let $A \in \mathbb{R}^{n\times n}$, $x \in \mathbb{R}^n$ where $x \neq 0$, and $\lambda \in \mathbb{R}$ then λ is an *eigenvalue* of A with an associated *eigenvector* x if and only if

$$Ax = \lambda x$$

Theorem 3.5.2 (Eigen Decomposition Theorem). Let $A \in \mathbb{R}^{m\times n}$ such that A^{-1} exists. Then the eigenvalues of A denoted $\lambda_1, \ldots, \lambda_n$ and eigenvectors $v_1, \ldots, v_n$ are unique and

$$A = PDP^{-1}$$

Where $P = \begin{bmatrix} v_1 & \cdots & v_n \end{bmatrix}$, and D is diagonal such that $d_{ii} = \lambda_i$.

3.6 Singular Value Decomposition

Theorem 3.6.1 (Reduced Singular Value Decomposition). Let $U \in \mathbb{R}^{m\times n}$ be orthonormal, $\Sigma \in \mathbb{R}^{n\times n}$ be diagonal, and $V \in \mathbb{R}^{n\times n}$ be orthonormal, then any $A \in \mathbb{R}^{m\times n}$ can be written in the form

$$A = U\Sigma V^T$$

This is known as a reduced singular value decomposition of A.

Theorem 3.6.2 (Eckart-Young-Mirsky Theorem). Let $A \in \mathbb{R}^{m\times n}$ such that $\text{rank}(A) = r \leq n$ and have a reduced singular value decomposition $A = U\Sigma V^T$, then for $k \leq r$ we define

$$A_k = \sum_{i=1}^{k} \sigma_i u_i v_i^T.$$

For all other matrices B such that $\text{rank}(B) = k$ we see for the two norm

$$\|A - A_k\|_2 \leq \|A - B\|_2,$$
$$\|A - A_k\|_F \leq \|A - A_k\|_F$$

and

$$\|A - A_k\|_2 = \sigma_{k+1},$$
$$\|A - A_k\|_F = \left(\sum_{i=k+1}^{r} \sigma_i^2 \right)^{\frac{1}{2}}.$$

4 Problem Solution

4.1 Problem Restatement

We wish to select the best fingerprints for a given set of atomic environments. This is equivalent to creating a matrix where the rows are atomic environments

and the columns are fingerprints and selecting the best columns of this matrix. In the language of linear algebra, this is equivalent to finding the optimal CUR factorization.

Definition 4.1.1 (CUR Factorization). Let $A \in \mathbb{R}^{m\times n}$ and $0 < k \leq n$, then the *CUR factorization* of A is

$$A \approx CUR$$

where $C \in \mathbb{R}^{m\times k}$ containing k columns of A, $U \in \mathbb{R}^{k\times k}$ such that $\text{rank}(U) \leq k$, and $R \in \mathbb{R}^{k\times n}$ containing k rows of A.

It can be shown that $UR = C^+A$, thus we only need to find the best possible C. To do this, we follow a modified approach of Boutsidis and Woodruff in [1].

4.2 Setup

For the rest of this paper, we let $A \in \mathbb{R}^{m\times n}$ such that $m \gg n$ and $\text{rank}(A) < n$ and we let $0 < k \leq n$. We first proceed by forming A_k from Theorem 3.6.2,

$$A_k = \sum_{i=1}^{k} \sigma_i u_i v_i^T, \tag{1}$$

$$= U_k \Sigma_k V_k^T. \tag{2}$$

We now define the matrix E such that

$$E = A - AV_kV_k^T.$$

Recalling from Remark 3.4.3 that $V_kV_k^T$ is the orthogonal projector onto $\text{col}(V_k)$ we see that E is the components of $\text{col}(A)$ orthogonal to $\text{col}(V_k)$.

4.3 Deterministic BSS Sampling

We want to deterministically sample $k/2$ columns of A following the method of Batson, Spielman, and Strivastava (BSS) to sparsify a dense matrix while preserving the spectral properties.

Theorem 4.3.1 (Dual Set Spectral-Frobenius Sparsification, Lemma 3.5 in [1]). Let V_k be defined from (1), then there exists a set of weights $s = \{s_1, \ldots, s_n\}$ where $s_i \geq 0$ and at most $k/2$ of which are non-zero satisfying

$$\lambda_k\left(\sum_{i=1}^{n} s_i v_i v_i^T\right) \geq \left(1 - \frac{\sqrt{2}}{2}\right)^2, \text{ and}$$

$$\text{Tr}\left(\sum_{i=1}^{n} s_i a_i a_i^T\right) \leq \text{Tr}\left(\sum_{i=1}^{n} a_i a_i^T\right).$$

We also see if we assemble the set of weights s to the $n \times k/2$ matrix S the above is equivalent to

$$\sigma_k(SV) \geq 1 - \frac{\sqrt{2}}{2}, \text{ and}$$
$$\|A^T S\|_F^2 \leq \|A^T\|_F^2.$$

To compute the $n \times k/2$ matrix S we take leverage scores of V_k forming a set $\pi = \{\pi_1, \ldots, \pi_n\}$ where we let v_i be the ith row of V_k and

$$\pi_i = \begin{cases} \|v_i\|_2^2, & c \text{ is a radial fingerprints} \\ \|v_i/m\|_2^2, & c \text{ is a bond fingerprint} \end{cases}$$

For $j = 1, \ldots, k/2$ we define the entries s_{ij} as

$$s_{ij} = \sqrt{\frac{1}{k/2} \max_i |\pi_i|}$$

and we update the set π such that for all $\pi_l < \pi_i$ we set

$$\pi_l = \pi_i - \pi_l.$$

After obtaining S we set $C_1 = AS$ and define $B = A - CC^+A$. From Remark 3.4.5 we see B is making the remaining unselected columns of A orthogonal to the already selected ones.

4.4 Adaptive Column Sampling

To select the remaining $k/2$ columns of A we use the adaptive column sampling technique from [1].

Theorem 4.4.1 (Adaptive Column Sampling, Lemma 3.9 in [1]). For $i = 1, \ldots, n$ define the discrete probability distribution where

$$p_i = \|b_i\|_2^2 / \|B\|_2^2,$$

and b_i is the $i$the column of B. Sample $k/2$ columns of A in $k/2$ i.i.d trials where each column has probability p_i to be chosen. Let C_2 be the $m \times k/2$ matrix of the $k/2$ columns sampled from A, then define $C = \begin{bmatrix} C_1 & C_2 \end{bmatrix}$. Then for any $k \in \mathbb{Z}^+$ we see

$$\mathbb{E}\left[\|A - CX_{opt}\|_F^2\right] \leq \|A - A_k\|_F^2 + \frac{1}{2}\|B\|_F^2,$$

where $X_{opt} \in \mathbb{R}^{k \times n}$ is the matrix of rank at most k and CX is the best rank k approximation to A in the span of C.

We follow this process for forming the matrix C_2 except we select the radial fingerprint columns with a higher probability. Specifically, we scale the columns of B corresponding to bond fingerprints by $1/m^2$ and then compute all of the required p_i.

4.5 Oversampling

Throughout this section, we have been treating A as if $\text{rank}(A) = n$. However, in practice, we observe that $\text{rank}(A) \ll n$, which is ultimately why we are using sampling rather than DEIM-like methods. Additionally, since there can be many similar columns, we must oversample with some parameter $k^* \gg k$ such that we can sample a variety of radial and bond fingerprints.

For each value of n, we take the first selected α_n to be the best one. We then find the m with the most β_ks selected, forming the set B. We look to find the set $\hat{B} \subseteq B$ such that

$$\hat{B} = \left\{ \beta_k \in B : \sum_i \sum_{j \neq i} |\beta_{ki} - \beta_{kj}| \text{ is maximized} \right\}.$$

In other words, we look to find the set of β_ks that span B the best.

5 References

[1] Christos Boutsidis and David P. Woodruff. Optimal cur matrix decompositions. *SIAM Journal on Computing*, 46(2):543–589, 2017.

[2] D. Dickel, M. Nitol, and C. D. Barrett. LAMMPS implementation of rapid artificial neural network derived interatomic potentials. *Computational Materials Science*, 196:110481, August 2021.

[3] Conrad Sanderson and Ryan Curtin. Armadillo: a template-based C++ library for linear algebra. *Journal of Open Source Software*, 1(2):26, June 2016.